July 1990 Final, Jan 85 – Jun 90

(U) The Information Distribution System: IDS – An Overview
1L162618AH80

WO# 44592002483101

Samuel C. Chamberlain

Ballistic Research Laboratory
ATTN: SLCBR–DD–T
Aberdeen Proving Ground, MD 21005–5066

BRL–TR–3114

With the advent of available computer technology, vast amounts of information can be moved and manipulated. Currently, information exchange between ”fighting level” forces (i.e., brigade and below) is constrained by relatively low–frequency tactical radios that do not support the high bandwidths common in most modern computer networks. Thus, a unique environment is evolving in which computer processing power grossly out performs communications power. The Ballistic Research Laboratory (BRL) Information Distribution System (IDS) is an experimental prototype that was implemented using a three–pronged approach to drastically reduce average bandwidth utilization (i.e., communications) at the expense of more processing: first, information is stored and exchanged in its most basic, primitive, and terse form (data abstractions of military concepts); second, a set of Information Distribution Commands are used to control the flow of information within and between tactical nodes so that only significant information is exchanged via the limited communications resources; and third, a connectionless communications protocol, named the Fact Exchange Protocol, is being developed that exploits the broadcast nature of combat net radio and provides very efficient communications.

Tactical information distribution, Low bandwidth communications,Automatic notification and distribution, Security Control, Data abstractions of military concepts, Information processing.

TABLE OF CONTENTS

LIST OF FIGURES

PREFACE

This is the first of a series of 12 reports describing the Ballistic Research Laboratory's Information Distribution System, or IDS, that was part of the Smart Weapons Systems (SWS) LABCOM Cooperative Program. The IDS was used in the SWS Program Demonstration held from 5 – 29 September 1989. For further information concerning the IDS contact Sam Chamberlain, or any member of the Information Concepts Team, at 301 – 278 – 6672 (DSN 298 – 6672). See also the: *Smart Weapons Systems (SWS) LABCOM Cooperative Program Summary Report*, by Harry R. Rogers and Major Andrew G. Ellis, BRL.

1. *The Information Distribution System: IDS – An Overview*, by Samuel C.Chamberlain.
2. *The Information Distribution System: The Factbase*, by George W. Hartwig.
3. *The Information Distribution System: Overview of DFB and SCM Technology*, by George W. Hartwig and Thomas A. DiGiacinto.
4. *The Information Distribution System: Network Monitor and Display*, by Thomas A.DiGiacinto.
5. *The Information Distribution System: The Fact Exchange Protocol*, *A Tactical Communications Protocol*, by Virginia A. Kaste.
6. *The Information Distribution System: The Working Map Application Program*, by Eric G. Heilman.
7. *The Information Distribution System: OPLAN: A Mission Planning Application Program*, by Eric G. Heilman.
8. *The Information Distribution System: A Scenario Driver*, by Eric G. Heilman.
9. *The Information Distribution System: ORG Chart: An Organizational Display Application Program*, by Kenneth G. Smith.
10. *The Information Distribution System: Load_dfb: An IDS Data Loader*, by Kenneth G. Smith.
11. *The Information Distribution System: Application Programmer's Interface Guide*, by Kenneth G. Smith.
12. *The Information Distribution System: IDS – Users Manual,* by Samuel C.Chamberlain, Thomas A. DiGiacinto, George W. Hartwig, Eric G. Heilman, Virginia L. Kaste, and Kenneth G. Smith.

I. OVERVIEW

With the advent of available computer technology, vast amounts of information can be moved and manipulated. Currently, information exchange between "fighting level" forces (i.e., brigade and below) is constrained by relatively low−frequency tactical radios that do not support the high bandwidths common of modern computer networks. Thus, a unique environment is evolving in which computer processing power grossly out performs communications power. The Ballistic Research Laboratory's (BRL) *Information Distribution System* (IDS) is an experimental software prototype that uses a three pronged approach to drastically reduce average bandwidth utilization (i.e., communications) at the expense of more processing: first, information is stored and exchanged in its most basic, primitive, and terse form (data abstractions of military concepts), second, a set of *Information Distribution Commands* are used to control the flow of information within and between tactical nodes so that only significant information is exchanged via the limited communications resources, and third, a connectionless communications protocol, named the *Fact Exchange Protocol*, was developed that exploits the broadcast nature of tactical net radio and provides very efficient communications.

The IDS has two major components: the *distributed factbase*, or DFB, and the *application* programs. The DFB serves as a repository and exchange medium for the information used by the application programs. Its function remains the same for any application program, only its configuration parameters change. Information is entered into, retrieved from, and manipulated within the DFB by the application programs that are connected to the DFB via high speed network connections (e.g., via Local Area Networks or internal network connection facilities). It is these sophisticated application programs that provide the tactical functions required at a particular node, e.g., maneuver planning, fire support coordination, air interdiction, antisubmarine warfare, or whatever. Since common DFB software provides the same services to all tactical systems (i.e., application programs), true information exchange is guaranteed and is highly automated. The tough job is the development of the primitive data abstractions that define how the information is stored within a DFB. It is now believed that a common, canonical set of abstractions can service most, if not all, tactical functions.

This report provides a functional description of the major concepts and design features of the IDS components. The DFB description includes: the *factbase* that stores information, the *security control module* that provides automatic dissemination and notification services, the *fact exchange protocol* that efficiently uses broadcast networks, and the *capability profiles* that provide the parameters that describe the standard operating procedures used to communicate between nodes. Application program are described that were developed to demonstrate and test the DFB and to provide an interface to the human operator. The DFB
also served several sophisticated applications programs that were provided by the other participants of the Smart Weapons Systems (SWS) LABCOM Cooperative Program.

II. INTRODUCTION

The Information Concepts Team of the Ballistic Research Laboratory's (BRL) Weapon Systems Technology Branch (WSTB) has developed an experimental Information Distribution System (IDS) to demonstrate several innovative information distribution concepts. The primary goal of the IDS project was to develop tactical computer science technology concepts that can support "fighting level" commanders and soldiers who must contend with highly dynamic, unpredictable, and hostile combat environments, and then, build an experimental prototype to further evaluate these concepts so that specifications can be developed for possible inclusion into future systems.

The Information Concepts Team developed a very simple model that it believes provides a natural decomposition of command and control (C2) tasks; **Figure 1** illustrates this model. All command and control tasks are divided into two very basic categories: battlefield management or information distribution. The *battlefield management* function consists of the myriad complex and sophisticated tasks that must be executed to win a battle, given that the proper information exists. The *information distribution* function provides the battlefield management function (or any application) with required information and insures that this information is available at the many locations at which battlefield management tasks reside. It is this second function, information distribution, that is the focus of this research effort. (Actually, this decomposition is applicable to systems other than just battlefield management; consequently, we give the battlefield management tasks the more generic title of simply *application programs*.)

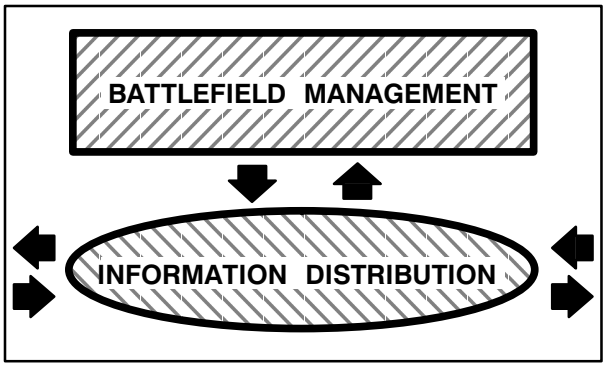

FIGURE 1: C2 Problem Decomposition

Two major philosophical tenets have driven the software architecture of the IDS: *first*, that this two part model of command and control provides an excellent, logical approach for implementing command and control systems, and *second*, a single information distribution system can serve *all* battlefield management functions, i.e., regardless of the military service or branch, provided that: one, equal emphasis on both the *computer* and *military* science aspects of the problem are considered, and two, worst case communication characteristics are handled. The software architecture of the IDS is founded on these basic tenets and the primary goal of the IDS is to provide an information distribution system that supports the requirements of most, if not all, battlefield management functions over unreliable communications channels.

Current tactical command and control systems support digital data exchange via formatted "messages" (character strings) or graphics symbols that must be interpreted by a human operator. These systems fall short in three key respects: first, the information is not in a form that computers can readily manipulate in a sophisticated manner (i.e., "understand"), second,

they use inefficient information exchange protocols that poorly utilize the limited bandwidth available on VHF–FM and HF–AM radios common to (and required by) lower echelon units, and third, these systems are still largely manual systems that require extensive amounts of user interaction for even the most mundane information transfers.

The IDS "fact oriented" approach to data distribution incorporates several new concepts in an effort to explore techniques that provide more flexibility and survivability to the information distribution function in command and control systems. These guiding concepts suggests the construction of *computationally* intensive systems, rather than *communications* intensive systems, that invoke information transfer only when internal computing fails to yield required results. A three pronged approach is being implemented to drastically reduce average bandwidth utilization (i.e., communications) at the expense of more computer processing: first, information is stored and exchanged in its most basic, primitive, and terse form (data abstractions of military concepts); second, a set of *Information Distribution Commands* are used to control information flow within and between tactical nodes so that only significant information is exchanged via the limited communications resources; and third, a connectionless communications protocol, named the *Fact Exchange Protocol*, is being developed that exploits the broadcast nature of tactical net radio and provides very efficient communications.

The main purpose of any information distribution system is to support the exchange of ideas or concepts. If a terse, computer efficient means of communicating between tactical nodes (i.e., units) is to be developed, the incorporation of *military* science as well as *computer* science is essential; this is perfectly exemplified by the process of defining abstract data primitives that reflect basic battlefield concepts (the first of the three pronged approach mentioned above). However, a significant effort must be expended "up front" as combat developers and computer scientists work together to build a canonical list of data abstractions describing basic military concepts; **Figure 2** demonstrates this approach.

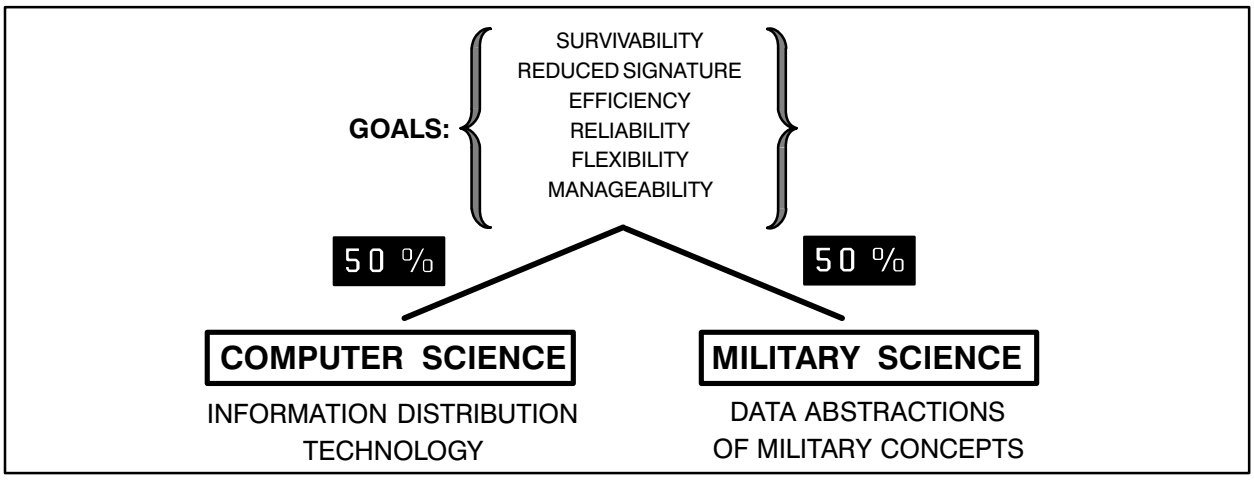

FIGURE 2: Equal Emphasis in Two Areas

Figure 3 shows the basic structure of the Information Distribution System, or IDS, software components. At the core of the IDS is the *distributed factbase*, or DFB, of which there would be one per tactical node (e.g., a vehicle). Residing "above" the DFB are several *application programs* that use the DFB to store and exchange information with other tactical nodes; battlefield management is performed using many such application programs. A DFB is composed of four conceptual modules: the factbase for information storage, the Fact Exchange Protocol (FEP) for inter–node information exchanges, a Package Protocol to enable connection between the DFB and application programs, and the Security Control Module (SCM) that controls access to the factbase and information flow into and out of the DFB. The *capability profiles*, or CAPs, contain the standard operating procedures that control information exchanges. In addition to the software, operational parameters contained in the CAPs are loaded into the DFB at start up time. Finally, reference material, i.e., information common to all DFBs such as weapon characteristics and Tables of Organization and Equipment (TO&E), is also entered into the DFB at start up time. **Figure 4** illustrates the

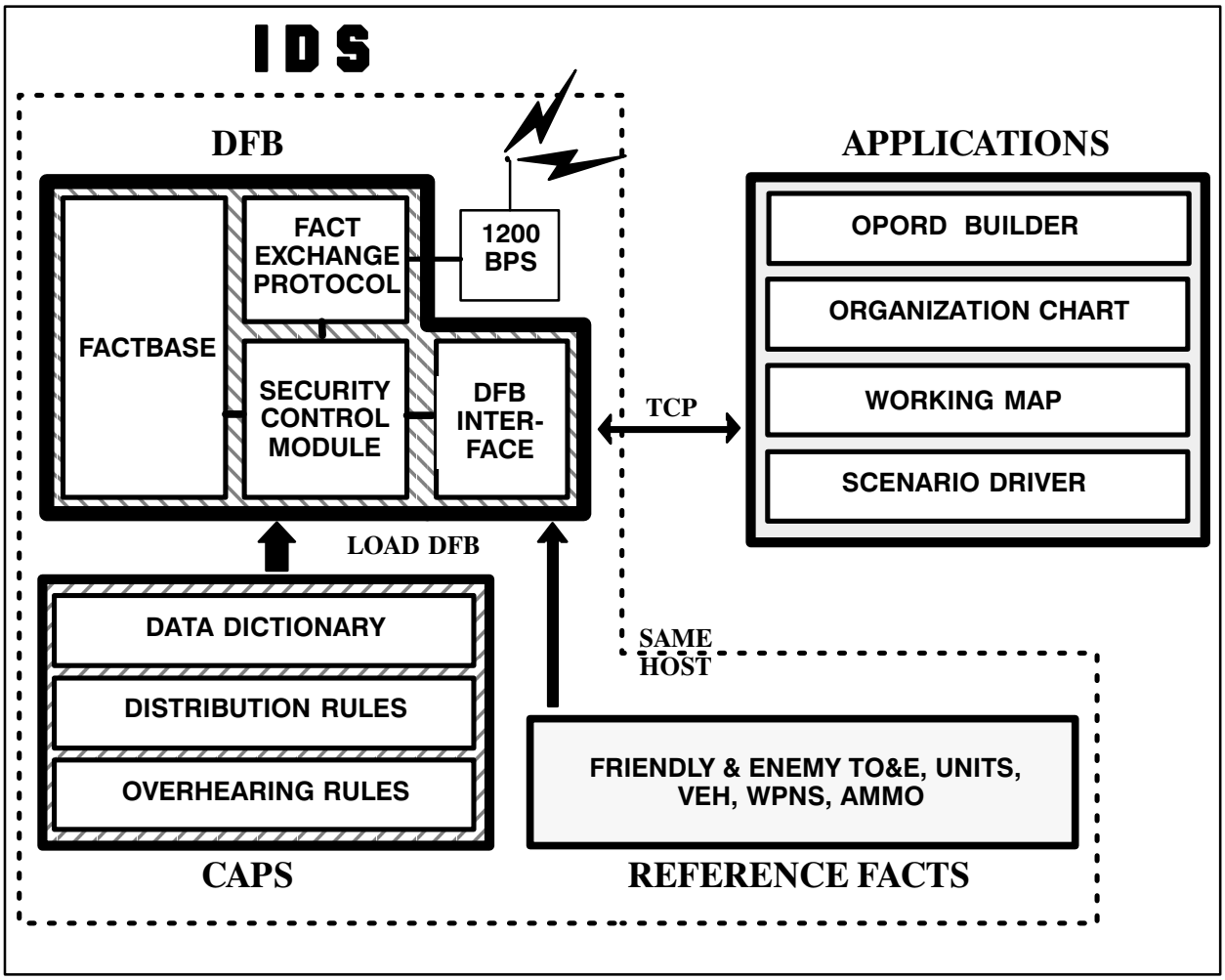

FIGURE 3: Block Diagram of IDS Software Components

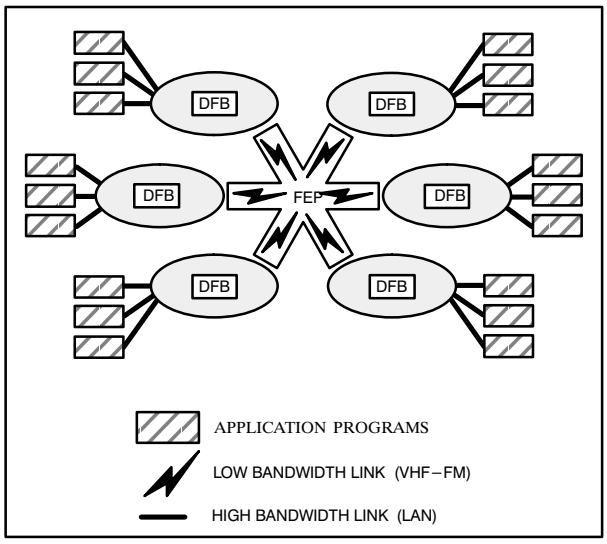

FIGURE 4: Several DFBs with Applications

two–part IDS software architecture using an example of six DFB nodes with associated application programs residing above them.

III. FACTS and FACTBASES

Information is stored within a RAM resident factbase as a collection of many interconnected *facts.* A fact is an instance of a pre–defined *fact–type*, or template, that is structured to describe an item, activity, or event common to the battlefield environment. A fact–type consists of a header and one or more *fact items*. The fact items will be one of five possible data types: integers, floating point numbers, character strings, references (to other facts), and lists (a collection of any of the above data types).

As facts are entered into the DFB (i.e., *stated to the DFB),* each is assigned a unique fact identification number, called a *fact id*, by the factbase. The unique fact id is permanently attached to the fact and moves with it as it traverses various DFBs so that the fact can always be uniquely referenced at any time or place; this is essentially what makes the factbase *distributed*. Currently, a fact id is an eight byte value that consists of the four byte Arpanet host address of the computer on which the fact was created and a four byte integer assigned by that resident factbase.

A direct comparison can be made between the structure of the factbase and a common relational database. A fact–type corresponds to a relation, a fact to a row of a database, and a fact item to a column of a database. The fact id would be a field common to all relations and would serve as the primary key for each row (entry) of the relation. Thus, relations can conveniently refer to entries in other relations via fact ids.

Every fact can be categorized as either, one, a dynamic fact, two, reference material, or three, a meta–fact. *Dynamic facts* describe changing battlefield events or activities, such as enemy sightings and control measures, and are stated and destroyed by the user via an application program. As just explained, every dynamic fact is associated with its factbase of origin (i.e., its host) by its fact id.

Reference material facts describe static "reference" information (e.g., Tables of Organization and Equipment (TO&E), vehicles, equipment, and ammunition) and have **static fact ids** that are common to *all* factbases. (Note: a special, reserved host address occupies the first four bytes of the fact id of a reference fact). This characteristic allows referral solely through the use of fact ids thus significantly reducing the amount of data that must be transmitted in many cases. Reference material facts are pre–loaded into each factbase during initialization and are never created unless

there is a doctrinal or equipment modification.

It is important to understand that static fact ids do not necessarily infer static information within the fact, although this is often the case. *Unit* facts are the typical example of this. A unit fact represents a specific military organization, such as the "2−11 Tank Battalion". Unit facts have static fact ids because they are never created by the user (only Congress can create units!); that is, only so many military units exist and these are all known in advance so that they can be stated ahead of time and assigned static fact ids for ease of reference. This does not imply that the information within the fact is static (e.g., a unit's location), but only that the existence of the fact is fixed. However, there are reference material facts with static information within the fact; typical examples are weapon characteristics and TO&Es; these facts have characteristics that will not change during a battle.

Meta−facts are facts about facts. A typical use of meta−facts is to represent potential modifications of other facts. For example, the unit facts in the factbase represent the "real world" situation as it is currently perceived. If one wants to recommend that a unit move to a new location, then a meta−fact would be used to represent this idea. Without changing the current location of the unit, one could state a meta−fact that refers to the existing unit fact but contains an alternative value for its location; the actual value of the unit's location is not changed. Therefore, a meta−fact simply contains a reference to another existing fact and a list of alternative values for fact items (fields) of the referenced fact. **Figure 5** illustrates this concept; in this example an alternate location for a unit is specified.

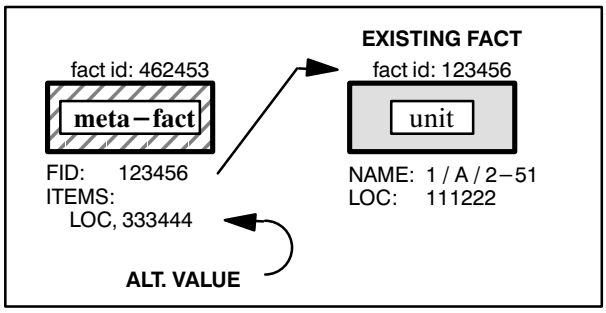

FIGURE 5: Example of a Meta−Fact

Figure 6 illustrates the current set of fact−types (canonical list of primitive data abstractions) and some of the relationships between them. Arrows indicate references (i.e., fact ids) to facts of other types. A brief summary of the basic set of fact−types follows; however, the complex selection process of these particular data abstractions is not a topic of this paper:

Fact−Type	Description
ammo	ammunition information
equip	equipment information
veh	vehicle information
org	typical organizations (TO&E)
unit	actual military units
line	geographic information
sensing	enemy sightings
target	to link units to sensings or lines
mission	operations order information
what_if	alternative fields to facts
commo	for inter−process communications

All information is stored as a fact to allow it to be easily exchanged between DFBs.

Figure 7 provides an example of several instances of facts used to describe standard units, organizations, vehicle, equipment, and ammunition (reference material). The task of designing the data abstractions is very intricate and requires a detailed knowledge of military operations and a broad systems perspective to main-

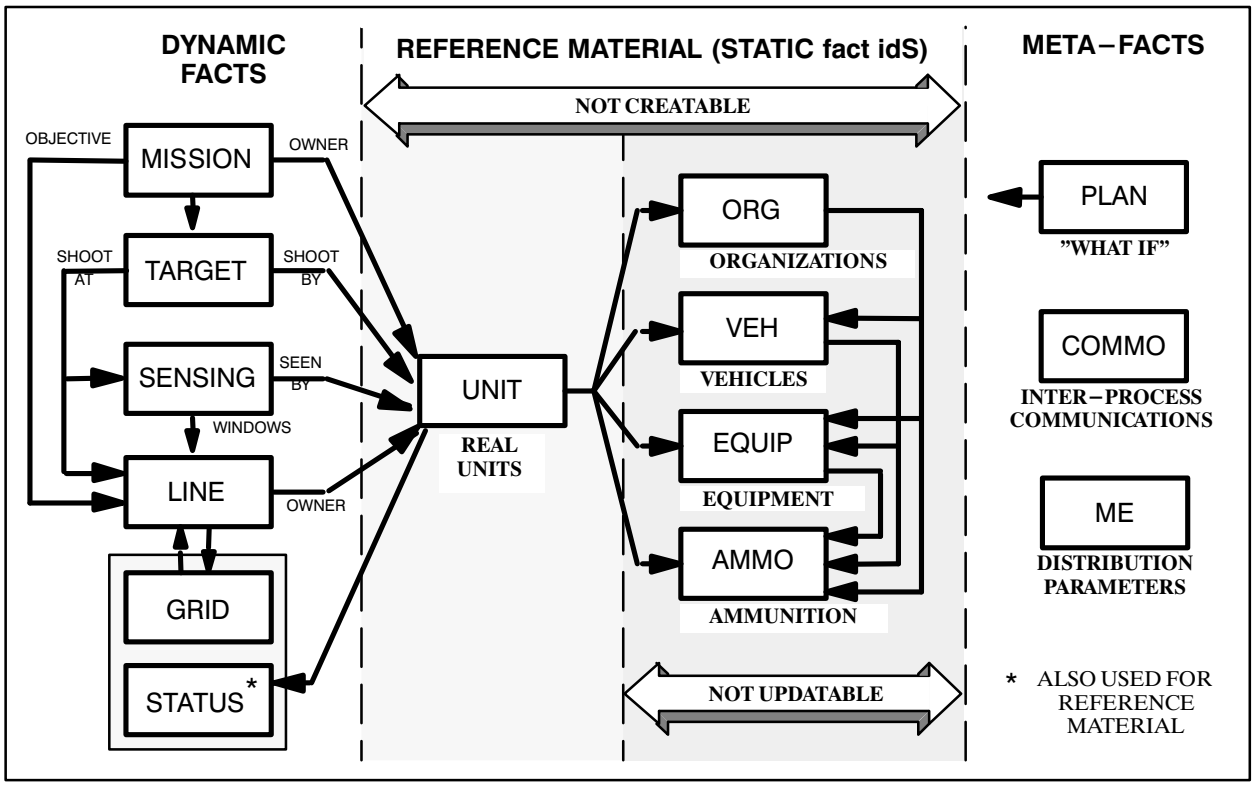

FIGURE 6: Data Abstractions (fact−types) and Some Relationships

tain the complex relationships between them. Figure 7 illustrates a very simple case; concepts such as targets and operations orders are much more difficult and involve considerable thought concerning what is really intended by these concepts.

There are six factbase basic commands:

1. **Define**: Create a fact−type (template)
2. **State**: Create an instance of a fact
3. **Update**: Modify an existing fact
4. **Kill**: Delete a fact
5. **Query**: Query the items of a fact−type
6. **Get_Fact**: Given an FID, get that fact

The *define* command creates a new fact−type, or fact template, such as those listed above (veh, equip, unit, etc.). The *state* command creates an actual instance of a fact, such as the 2−11 Tank Company (a unit fact), M2 Bradley (a vehicle fact), or an M68 cannon (an equipment fact). The *update* command modifies the items of an existing fact; a fact id is required in this command to identify the fact that is to be updated. The *kill* command deletes a fact from the factbase; it too requires a fact id.

The *query* command has been implemented with the worse case in mind: a query over a low−bandwidth communications channel. Since the information returned from a query may be voluminous, it is implemented as a two step process. The result of a query is not the facts themselves, but rather a list of the fact ids of those facts that satisfied the query. Once the fact ids are received, an application program executes a *get_fact* command for only those facts (or items) that are required. This technique allows the application programs to minimize information retrieval since many of the facts that satisfied the query may already be in the local factbase. Similarly,

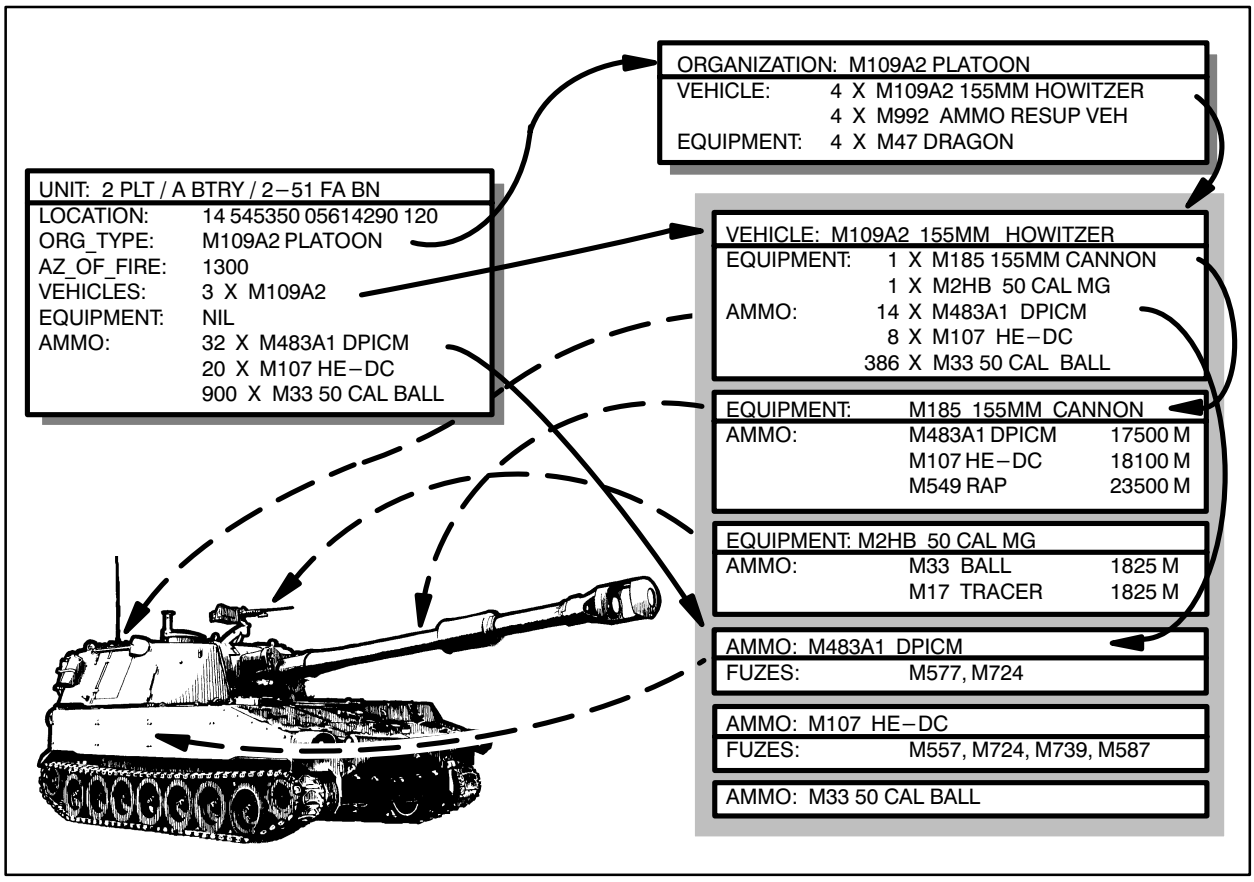

FIGURE 7: Typical Data Abstractions (fact−types)

queries are required only when fact ids are not known. For example, a get_fact can be used directly to retrieve a unit's location since unit facts are reference material so their fact ids are common knowledge to all DFB.

IV. SECURITY CONTROL MODULE

Ultimately, the SCM will provide four major tasks: one, to determine when local information is significant enough to be transmitted to remote DFBs (i.e., control information dissemination), two, to determine if received fact information should be entered into the resident factbase, three, to determine whether to honor requests for information (from either remote DFBs or local application programs), and four, to adjust FEP parameters (e.g., window size and number of retries) based upon communication channel measurements reported from the FEP (e.g., average acknowledgement round trip times). The focus of the initial prototype was on the first of these tasks.

The second of the three pronged approach is to control the flow of information within and between tactical nodes so that only "significant" information is exchanged. An important concept implemented in the DFB is the ability to automatically initiate pre−defined actions upon the reception of new information, whether the information comes from other DFBs or local application programs. The mechanisms that enables a DFB to execute automatic actions are named *information distribution commands,*

or IDCs. An IDC is a rule that describes what to do when certain information arrives. An IDC is composed of a *criteria* part and an *action* part.

DFBs exchange factbase commands, the most common of which is the update command. An update command is composed of four parts:

e.g., ''update fact−ID fact−type(optional) fact−item−list''

1. The code for ''update'';
2. The fact id of the fact to update;
3. An optional fact−type of the fact to be updated; and
4. A list of fact−items with new values to be updated.

The fact−type is required when a *new* fact is sent to a another DFB. The *state* command is used to create a new fact, and in return, a new fact id is assigned to the the new fact. For control and security reasons, applications are only allowed to create facts on a local DFB. When a new fact is sent to another DFB, it is sent as an update. In this case, the fact−type of the new fact is also sent so that the receiving DFB can interpret the fields. In fact, a new factbase command, ''remote−update'' (or ''rupdate'') was created to explicitly handle this situation. Most of the ''messages'' exchanged between DFBs are *update* or *rupdate* factbase commands.

The *information distribution commands* have the following structure:

"if **fact_type criteria** then **actions**" where

fact_type is the name of a fact definition,
criteria defines the comparison parameters for incoming factbase commands, and
actions execute instructions should the criteria be matched (e.g., SEND or NOTIFY).

A major area of study is the determination of what capabilities the criteria and actions of the IDCs must support.

Whenever a factbase command enters the DFB, whether from another DFB or from a local application program, its values are used for comparison in the criteria part of each IDC (currently active within the SCM of the DFB). If the IDC criteria is met, then the action part of the IDC is executed. There are currently two basic actions that can be executed: *send* and *notify*. Send transmits information to another DFB while notify tells an application program that an IDCs criteria has been met. IDCs with send actions are named *distribution rules* since their purpose is to control the information leaving the DFB. IDCs with notify actions are named *triggers,* and their purpose is to provide an automatic notification capability to the application programs using the DFB. Application programs can state triggers via the SCM, but not distribution rules; distribution rules strictly controlled via a privileged entry into the SCM. Triggers are implemented slightly differently from distribution rules as they compare the IDC criteria to the fact information *after*, rather than *before*, the information is entered into the factbase.

Distribution rules (i.e., a set of rules governing information distribution to other nodes) are tailored to each individual DFB. These distribution rules insure that only "significant" data (as defined by the commander and staff) are transmitted. For example, it is normally not significant to report

every round of ammunition that a tank platoon fires. More appropriately, ammunition status should be reported when it reaches pre–designated values or rates.

A very significant ramification of this distribution technique is that, theoretically, if the distribution rules are defined correctly, **a DFB should rarely, if ever, have to query another DFB for information**. This is not as strange as it may initially seem, in fact, it is the mode in which military operations are normally conducted. Commanders do not constantly call up their subordinates to ask if they've seen anything (i.e., query them), but instead, give the subordinates instructions ahead of time as to what to report (i.e., distribution rules). Thus, DFB to DFB queries should be rare occurrences provided that careful thought has gone into the distribution rules ahead of time. If more information is required by a node, then the distribution rules should be dynamically updated at the nodes that hold the required information.

IDC criteria involves four basic categories of values (i.e., variables): one, the values within the incoming factbase command (i.e., the new value), two, the values currently in the factbase, three, the last transmitted value (also stored in the factbase), and four, constants. The following examples demonstrate these four categories. A variable named "easting" contained within a ''unit'' fact with a fact id of 123456 is used in these examples:

A. Incoming factbase command:
''update 123456 easting, 45423;''
New easting of fact 123456 is: 45423

B. Current easting of fact 123456 is: 45380

C. Last sent easting of fact 123456 is: 45320

D. Constant (within an IDC): 45400
''if unit_type (easting > 45400) then...''

The value of 45423 in A is a new value for the fact–item named "easting" and would arrive as an incoming factbase command (e.g., reported by a subordinate unit). The easting value 45380 in B is the value currently stored in the factbase. The easting value 45320 in C is the value that was last transmitted (i.e., reported) to another DFB, such as a parent unit; this value is stored in a special location by the SCM called a pseudo–fact (which will be explained shortly). Finally, the value 45400 is a constant defined within an IDCs, such as a trigger stated by an application program.

The criteria portion of an IDC may execute comparisons between any of these categories of values: the incoming factbase command values, the current factbase values, last transmitted values, or constants. In addition, certain parameters about the incoming factbase command itself are also available for IDC criteria processing. These include the factbase command's:

- source (the DFB it's from or ''local'')
- destination (the DFB it's to)
- type (e.g., state, update, kill, etc.)
- time of arrival
- fact id

The following examples of criteria expressions demonstrate the use of these various types of values. A prefix is applied to the variable name to identify its category; these are given in the legend.

EXAMPLE A states: *let me know when a unit's easting changes by more than 50 meters*. This IDC criteria will "fire" whenever a location update arrives in which the instan-

LEGEND for fact variable (name) prefixes:
Current factbase value: none Incoming fact command value: ''t_'' Last transmitted value: ''l_'' Constants: none (quotes for strings) Fact command value: ''m_''
EXAMPLES: A. (t_easting − easting) > 50 B. (t_easting − l_easting) > 100 C. (t_easting) > 45423 D. m_src == "cdr/2−11" E. m_src == "local" F. m_fid == 123456

taneous change is greater than 50 meters from the current factbase value.

On the other hand, EXAMPLE B states: *let me know when a unit's location changes by more than 100 meters from the last time I reported it to someone else*. This IDC criteria is very useful in filtering status reports up the chain of command since brigade commanders do not necessarily want to know the same level of detail as battalion or company commanders.

EXAMPLE C simply states: *let me know when a unit crosses east of the 45423 grid line*. EXAMPLE D fires on incoming factbase commands from the commander of the 2−11 Tank Battalion while EXAMPLE E fires when a factbase command arrives from one of the application programs residing on this DFB. Finally, EXAMPLE F fires whenever a factbase command concerning fact 123456 arrives. Of course, several expressions may be combined together using logical operations like AND and OR to produce quite powerful criteria.

In addition, specialized functions can be implemented. Three current examples are: *exists,* that check for the existence of a specific factbase variable name (i.e., a *fact item*) within an incoming factbase command; *abs*, that computes the absolute value of an expression; and *distance*, that uses both the "easting" and "northing" fact items (i.e., cartisean map coordinates) contained in several fact−types to compute overall movement changes rather than just easting or northing moves.

Once the criteria portion of a IDC fires, the action portion of the IDC is executed. As previously explained, there are two basic actions: send to another DFB and notify an application program. (IDCs with the send action are called distribution rules, and IDCs with notify actions are called triggers.) Distribution rules can send several different entities:

1. the complete factbase command.
2. an item in the factbase command.
3. the complete fact referenced in the factbase command.
4. fact item(s) from the fact referenced in the factbase command.
   a) the value itself.
   b) if a reference to another fact (i.e., a fact id), the fact it references.

These features provide a powerful capability to automatically send information to other DFBs as the following examples demonstrate:

1. SEND MSG TO A/2−11
2. SEND MSG.easting TO B/2−11, C/2−94
3. SEND FACT TO SUB_UNITS

4a. SEND FACT.easting TO PARENT

4b. SEND FACT.@target TO ADJACENT_UNITS

EXAMPLE 1 states: retransmit the same incoming factbase command to A Company / 2−11 Tank Battalion.

EXAMPLE 2 states: retransmit the factbase command with just the expression con-

cerning the unit's grid easting to two companies: B/2−11 and C/2−94.

EXAMPLES 3 and 4 demonstrate that special variables are available that store specific sets of common addresses based on the current task organization. These variables are: PARENT, for one's parent unit, SUB_UNITS for one's subordinate units, and ADJACENT_UNITS, for one's siblings in the task organization tree plus any other required units.

EXAMPLE 3 states: transmit the complete fact referenced by the incoming factbase command to my current subordinate units.

EXAMPLE 4a states: transmit just the easting fact−item of the fact(s) referenced by the incoming factbase command to my current parent unit.

EXAMPLE 4b demonstrates a single level of indirection. It states: transmit the complete fact referenced by the target item of the fact referenced by the incoming factbase command. This capability is very useful for the many cases in which facts referenced other facts that contain important additional information.

Trigger actions (notify) are different from distribution rule actions (send). Triggers are stated by application programs to allow them to be automatically notified when certain information arrives from another DFB or application program. When an application program states a trigger, it provides a *handle* (a unique name) along with the trigger criteria. The SCM attaches the application programs process ID to the trigger. When the trigger criteria is fired, the SCM notifies the owning process that one of its triggers fired and provides the handle of the trigger and the fact id of the fact that caused the trigger to fire. It is then the task of the application program to retrieve the fact and do what it pleases with it.

As mentioned earlier, there are *pseudo−facts* that are used by the SCM to store additional information about facts. Pseudo−facts are used to temporarily store current factbase information when it is required to evaluate a rule. Typical examples include: the last time the fact, or an item from the fact, was transmitted to another DFB and the additional factbase command information previously described (e.g., its source or destination). There are several other information items under consideration for storage in pseudo−facts, such as application layer fragmentation of information. The uses and requirement of pseudo−fact are a key area of ongoing study.

Because the same distribution rules may reside in several DFBs (normally in adjacent DFBs), infinite loops or redundant copies of messages can easily propagate. For example, distribution rules that exchange information between adjacent units can bounce facts back and forth forever. To prevent this situation there are four rules that are always in effect:

I. Do not send a message back to its source.
II. If a message is from a parent, do not send it to an adjacent unit.
III. If a message is from an adjacent unit, do not send it to anyone.
IV. If a message is from a sub−unit, do not send it to another sub−unit.

A synopsis of these rules is given below. **Figure 8** illustrates a hierarchal (tree) struc-

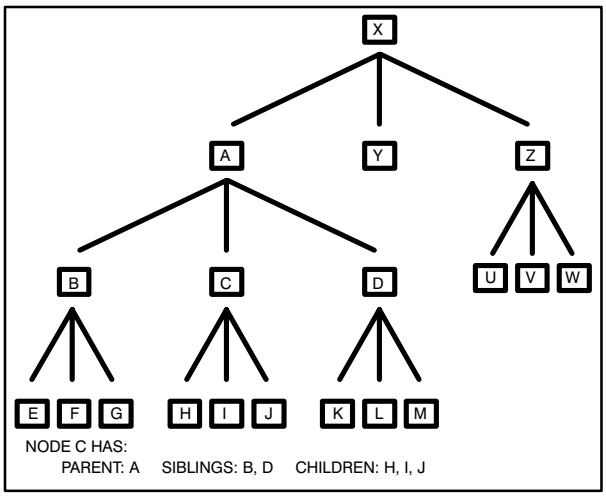

FIGURE 8: Typical Tree Structure

ture common in military command and control. The definition of an *adjacent* unit is *any* unit that is not a parent or child, thus Y can be an adjacent unit of C, and vice versa.

Rule I is self explanatory.

Rule II states that the distribution of information down the chain of command is the responsibility of the parent nodes, and subordinates will not have to exchange this information with each other; this is the common policy in present command and control procedures.

Rule III states that information from adjacent units, that is, units not in the chain of command, is never forwarded. If a unit wants to pass information to an adjacent units, it will do so directly and will not depend on the adjacent units to re−address the information and pass it on. In essence, this prevents the application layers of network protocols from re−addressing and forwarding packets. This is NOT the same as forwarding a packet (without re−addressing it) at the network layer, which is the routing function in packet switched systems, such as packet radios. Likewise, information from adjacent units will reach one's parent node only from the adjacent units. For example, if K sends information to J, the only way C should receive this same information is via D, K's parent. Thus, information is allowed to be forwarded across the chain of command only once. Of course it can be modified locally and resent, but this is a new message.

Rule IV states that a parent unit is not responsible for distributing information from its sub−units to other sub−units. In other words, sibling units are responsible for passing information directly between each other and their parent unit. (Of course, the parent may pass information to its parent, siblings, and adjacent units.) This is a big change from present command and control policy, but it significantly reduces the number of information transmissions required when combined with the appropriate protocol (i.e., the Fact Exchange Protocol) that supports IDS features such as multicast addressing.

These rules prevent infinite loops from occurring and permit the implementation of innovative new information distribution policy. Briefly stated: *when passing information up the tree, also include any sibling and adjacent units that require the information*. This scheme propagates information as a "wave" up the echelons allowing the units that are most effected by the information to obtain it first. Thus, a tank section may very well send sensing data to adjacent tank sections even though they belong to different tank companies, battalions, or brigades because the information may significantly effect their neighbors. For example, in Figure 8, J might send sensing information to four other nodes: its parent C and

three adjacent nodes – H and I (its siblings) and K (an adjacent unit), thus crossing the standard chain of command. K can not forward the information *to anyone* since the information came from an adjacent node and doing so could cause an infinite loop of information. However, once an application program, located on K's DFB, modifies the incoming fact, it may then be sent to other nodes.

One issue to be addressed is the procedure for passing information to a unit on a different network. This is a network layer [3,6] routing problem that is handled by standard technologies such as packet radios and gateways. Distribution rules should not be used to convert a DFB into a gateway except under unusual and tightly controlled circumstances. However, a dozen inexpensive, mobile gateways could be allocated per brigade to connect various VHF–FM radio nets thus providing convenient exchange of information between hosts on different networks.

V. FACT EXCHANGE PROTOCOL

Information exchange between DFBs is supported by the Fact Exchange Protocol, or FEP; this is the aforementioned third approach to drastically reduce average bandwidth utilization. The FEP is a transport layer protocol [3,6] that provides connectionless, reliable datagram service, but it is specifically tailored to do this over very unreliable, low–bandwidth, combat net radios. The FEP provides three new features: one, arbitrary multicast addressing and broadcast, two, the overhearing of all transmissions on communications channels, and three, an emission control (EMCON) mode of operation (i.e., listening silence). Like any Open Systems Interconnection (OSI) transport protocol, it can be encapsulated within in the standard DoD Internet Protocol (IP), but a small network layer protocol has also been developed to support multicast addressing and overhearing. The focus of the initial implementation centered on reliability with terseness utilizing the overhearing, multicast, and other techniques to minimize radio transmissions.

Recall that each incoming factbase command, whether from a local application program or another DFB, is evaluated against the criteria portion of each IDC by the SCM. When a distribution rule (i.e., an IDC with a *send* action) fires, the SCM executes the send action of the IDC and passes information to the FEP. This information will include one or more factbase commands (often including the one that caused the firing of the rule), destination address(es) or a broadcast indicator, and other transmission parameters as listed in the send action.

The FEP converts each factbase command into a *fact exchange*, or FEX, which is the unit of reliable transfer between FEP transport layers (i.e., the Transport Protocol Data Unit, or TPDU, in OSI Reference Model terminology[4]). This means that each fact exchange (containing a single factbase command) is acknowledged when reliable data transfer is indicated. Reliable transfer is inferred by providing one of more destination addresses to the FEP. By definition, broadcast fact exchanges do not require reliable reception.

In the past, commanders and their staffs have kept themselves informed by simply listening to overheard voice transmissions occurring on several radio nets. Similarly,

the collection of “free” digital data (at no cost in bandwidth) is a feature of the FEP. The capability to collect overheard information is provided to reduce unnecessary retransmissions that consume bandwidth. Thus, datagrams received by the FEP are put in one of two main categories: datagrams addressed to the resident DFB and overheard datagrams. Only datagrams meant for the resident DFB are acknowledged, but all received datagrams are forwarded to the Security Control Module (SCM) for processing. In the initial implementation of the SCM all overheard fact exchanges are entered in the factbase. However, upcoming versions will use IDCs to determine the pertinence of overheard information for entry into the factbase.To allow the overhearing ability over standard local area networks, or LANs (e.g., Ethernet), the User Datagram Protocol (UDP)[2], a standard DoD protocol, was used in broadcast mode to encapsulate the FEP datagrams.

Overheard datagrams must be meaningful. However, standard network protocols, like the DoD IP, may arbitrarily fragment packets when they exceed the maximum transmission unit size (MTU) of the datalink protocol. Arbitrary packet fragmentation can cause a comprehension problem since information describing a fact may arrive in incomplete, meaningless pieces. Therefore, fragmentation must occur at a logical, or meaningful place in the information so that the overheard pieces make sense. This requires that the Security Control Module handle any fragmentation, not the FEP. This also requires that the MTU be passed up to the SCM (an OSI reference model application layer [3,6]) so that the fragment size is known for each communications channel. For this implementation, the channels were Ethernet and FSK (frequency shift keying) modems for VHF–FM radio, both with MTUs of approximately 1500 bytes. Although unlikely, facts larger than the MTU will have to be divided at logical internal boundaries by the SCM before being passed to the FEP.

Conversely, several fact exchanges may be packed into a single MTU sized packet, called ‘‘blocking’’[4], to reduce bandwidth utilization. A significant reduction in bandwidth usage is achieved by reducing the number of radio transmission because radio transmissions usually require a ”preamble” that is often relatively large in comparison to the size of the information packet. Making each packet as large as the MTU minimizes the number of transmissions required to send information, which in–turn, minimizes the effect of the radio preambles. Therefore, a ‘‘packet’’ no longer has a single destination (i.e., host address) associated with it since every fact exchange contained within the datagram may have a different destination; this is illustrated in **Figure 9** (see also List of Acronyms).

FEP datagrams are sent between hosts, or groups of hosts, called a *host set*, for multicast addressing. A datagram can contain several fact exchanges, each with its own header that contains a 32 bit fact exchange identification number. Fact exchanges, *not* datagrams, are acknowledged. The connectionless nature of the FEP required a selective (out of sequence) acknowledgment scheme for each fact exchange entity. The FEP decomposes incoming packets into separate fact ex-

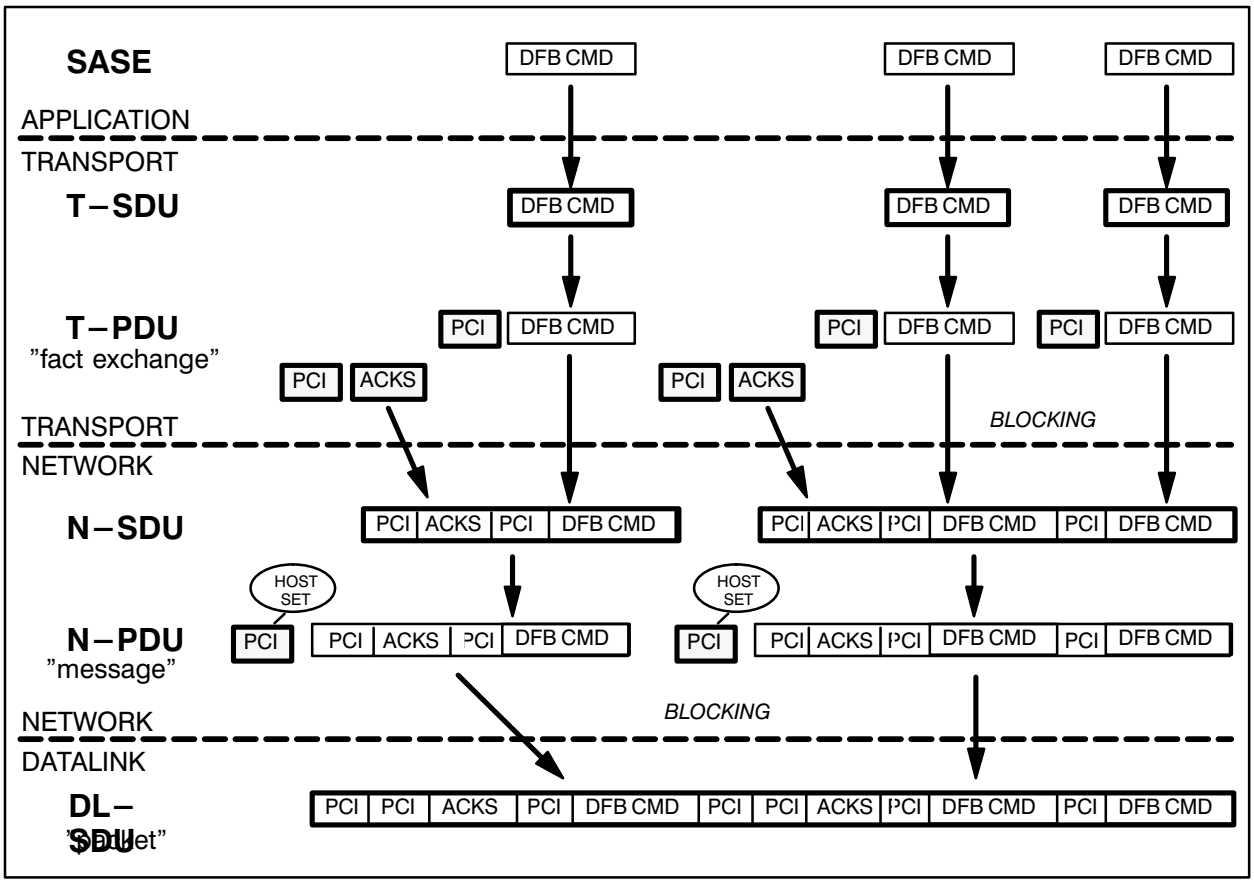

FIGURE 9: FEP Features in the OSI Reference Model Terms

changes and returns an acknowledgement for those fact exchanges intended for the resident host. It then tags all fact exchanges as either intended for the resident factbase or overheard and passes them to the SCM for further processing.

In addition, the FEP keeps track of the hosts with which it is allowed to exchange information; the FEP sends acknowledgements only to hosts that are listed in its authorization table. Consequently, incoming fact exchanges (from other DFBs) addressed to the receiving host are further tagged as either ''authorized'' if in the table or ''unauthorized'' if not. Thus, being entered into the table is the DFBs way of ''establishing a connection.'' In subsequent versions of the FEP, unauthorized fact exchanges may automatically trigger action by the SCM to evaluate the source DFB of the unauthorized fact exchange for entry into the authorization table. This would be somewhat analogous to dynamically establishing a connection.

Figure 10 illustrates the FEP datagram headers for both the transport and network layers; the transport layer also includes the individual fact exchange headers. White areas contain the principle header information; shaded areas are the individual fact exchange headers; and the hashed areas represent data (i.e., factbase commands).

Due to the wide variation in communication channel bandwidths (e.g., 10 Mbps LAN and 1200 bps FSK modems over the VHF−FM radios), flow control concepts

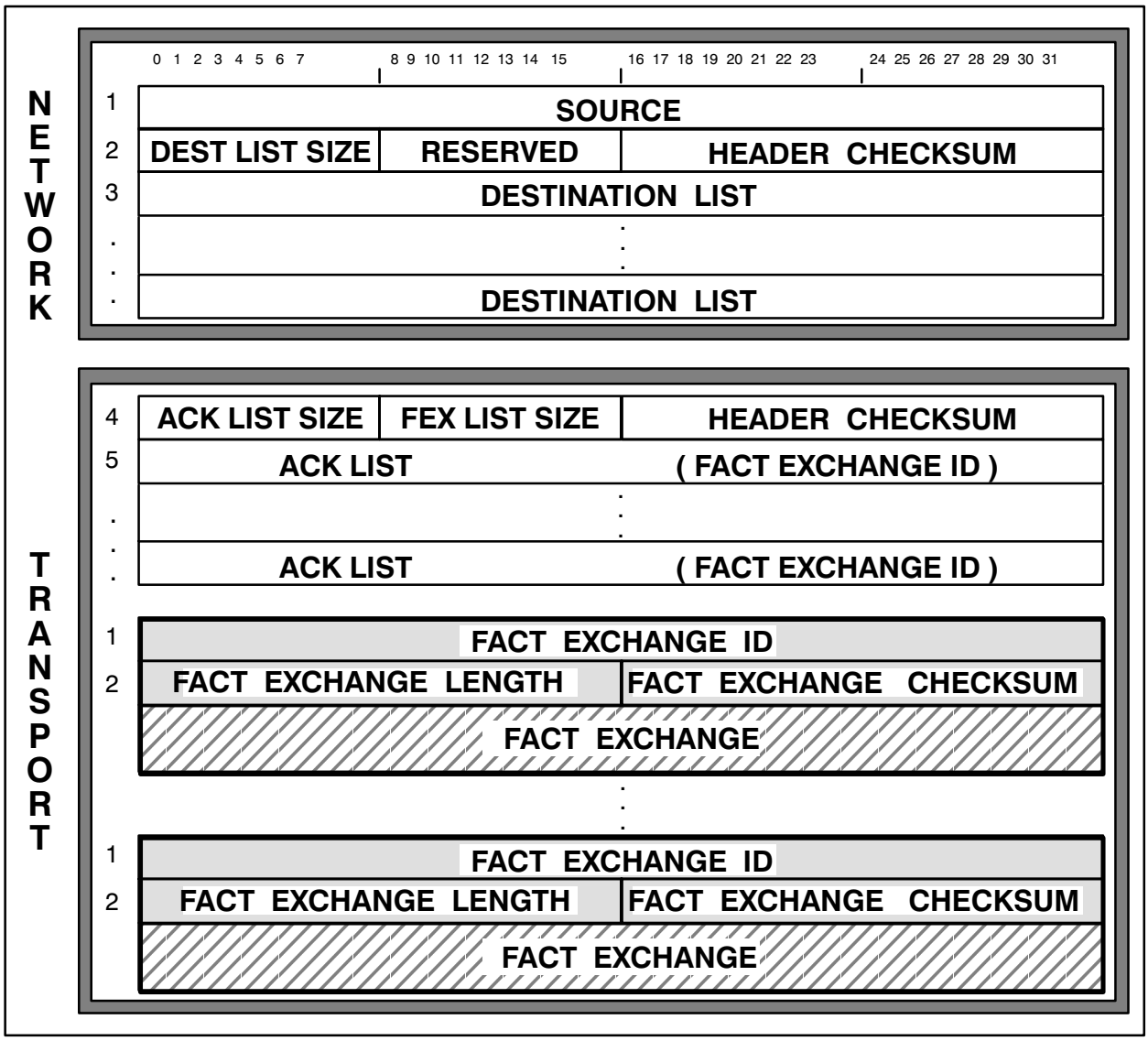

FIGURE 10: Fact Exchange Protocol Header (by layers)

must be re−examined. For example, parameters such as ‘‘window’’ (i.e., the allowed number of outstanding unacknowledged fact exchanges), ‘‘wait time−out’’ (i.e., time to wait for an acknowledgement before retransmission), and the number of retries (before giving up) are based on a priori knowledge of the destination host and communication channel being used. These parameters are defined for both channels and hosts; however, it is typically the low bandwidth communications channel and not the hosts that limit information exchange. Since a single−hop network is the communications medium for this system (rather than an internet), these network parameters are relatively easy to obtain. These parameters can also be set explicitly in the distribution rules to be passed to the FEP. Later versions of the FEP may provide the SCM with fact exchange timing measurements to be used to dynamically adjust these parameters.

Future plans for the FEP include the implementation of an EMCON, or radio silence, mode of operation. This would emulate the common voice communications practice of silently listening to radio nets, even for messages intended for the listener. A node in EMCON mode would receive information without returning acknowledgments. Upon discontinuance of EMCON mode, a "bulk" acknowledgment could be sent to update only the most recently received fact exchanges (for a particular fact). Actual transmission of the bulk acknowledgement could be accomplished through the use of any appropriate media (e.g., radio, floppy disk and motorcycle, etc.). This would be predominantly a SCM function, but there are a few FEP capabilities that must be implemented to support EMCON operations.

There could be two types of EMCON mode transitions: *graceful* and *crash*. In graceful mode, information currently in the FEP buffers would be sent before EMCON mode is entered. In crash mode, a panic clearing of all FEP buffers would be executed, and transmissions immediately halted.

A typical use of EMCON mode would be to help mask the movement of command posts (CPs). Either the CP or the alternate CP would be in EMCON mode when it is not serving as the primary CP. Once the alternate CP is ready to assume the duties as primary CP, it would come out of EMCON mode, and the previous primary CP would go into EMCON mode. As the two CPs exchange control, they would enter and leave EMCON mode thus electronically masking their moves. In addition to radio, bulk acknowledgements could easily be executed via a messenger on a motorcycle carrying a small "floppy disk" or a memory chip. EMCON mode could also be used by the commander to enforce radio discipline via remote commands.

VI. DFB INTERFACE TO APPLICATIONS

Standard TCP/IP sockets (standard DoD protocols) [2] serve as interfaces between applications and the factbase. This allows the applications to reside either on the same host as the DFB or on a separate processor connected by a reasonably reliable data link medium (e.g., a LAN or a common bus). DFBs exchange information with other DFBs over unreliable, low–bandwidth data links using the FEP; however, application programs are designed (and expected) to exchange information with DFBs over reliable, high–speed links.

This design provides significant flexibility to both the software and hardware architecture. Because a standard protocol is used between the applications and the DFB, there is absolutely no requirement that the application processors be similar. As was well learned from the Arpanet experience, this architecture allows totally different processors to be inter–connected. Thus, highly specialized processors can be built for specific applications that share a common set of hardware and software running the DFB software (this is not technically necessary, but is more realistic). Of course, TCP/IP sockets can be established within the *same* processor if desired.

A good example of a requirement of this this flexibility is the dispersed command post. It may have a single DFB connected to

several dispersed application programs (each running on their own processor) via TCP/IP sockets over a LAN , or separate DFBs connected via a LAN using the FEP.

To facilitate the establishment of connections between application programs and the DFB, the "package protocol" utility was used. It handles most of the work in establishing the TCP/IP sockets for application program developers. The "package protocol" is available on any machine supporting the standard TCP/IP protocols and the "C" programming language.

VII. CAPABILITY PROFILES (CAPS)

Node initialization requires that the previously described information and constraints be entered into the SCM of each DFB. This information is stored in *capability profiles*, or "CAPs", that describes the standard operating procedures (SOP) for information distribution of that unit. Eventually, the CAPs, like all other information, will be stored as facts to facilitate information exchange. Three types of parameters are currently included within the CAPs; 1) a *data dictionary* that defines every valid fact–type acceptable to the factbase, 2) the *Information Distribution Commands* that control the flow of information on the network, and 3) the *authorization table* that describes with who the DFB is allowed to communicate.

The CAPs can be modified dynamically (even from remote locations) allowing all the items previously mentioned to be maintained as best appropriate for a particular situation. For example, several sets of CAPs could be developed for various tactical situations, such as offensive or defensive postures, special operations, or training scenarios. The set of CAPs would be stored locally at each DFB (i.e., host) and loaded on command. As battle conditions change, different CAPs could be loaded into the SCM thus easily adjusting the operating conditions for the DFB. It is envisioned that the CAPs, which basically implement the SOPs for a unit, should be developed by experts on doctrine and tactics (e.g., US Army Training and Doctrine Command schools) and modified by specialists in individual units; the CAPs would not normally be modified by the soldier during combat.

The CAPs can store many types of SOP information. A good example is a special variable, called *reporting depth*, that is used to implement automated status reporting. Reporting depth provides a short–hand technique to define and control the amount of status information reported. Reporting depth dictates the number of echelons below your own about which information is received. For example, a reporting depth of two is traditionally used by commanders. Therefore, every unit should have information about units two echelons below in the command structure (e.g., brigades monitor company status, battalions monitor platoons, etc.). In this case, each unit sends status information about its direct subordinate units to its parent unit (e.g., battalions send information about their companies to their parent brigade). Thus, a brigade would receive status information about all its companies, and if a status report about its battalions is required, it is computed using the information received about its companies.

Another special value is the unit's *distribution envelope.* This describes the unit's

information purview based on the reporting depth and any other special cases. It is basically a list of those units with which a DFB expects to exchange information and may be dynamically modified based on the desires of the unit commander. In a low bandwidth tactical environment, an increase in the distribution envelope will decrease the usable bandwidth available to other units and produce a larger electronic signature. The distribution envelope concept is also used to control the entry of overheard information since, for obvious security reasons, it would not be wise to incorporate all overheard information (e.g., should a node be compromised or captured). Thus, any control information may be stored in the CAPs, and its contents are controlled entirely by the commander.

VIII. APPLICATION PROGRAMS

Above the DFB resides the battlefield management functions that execute the myriad of command and control tasks. Generically, we call these *application programs* since any type of application may be implemented. Since one cannot "see" a DFB, application programs were developed to demonstrate the capabilities of the system and to provide an interface to the DFB for testing and evaluation purposes. Since application programs isolate the user from the workings of the DFB, they also assist in controlling access to the information stored within a DFB.

Information (facts) can be retrieved manually using queries or automatically using DFB triggers. New information may be entered by stating new facts or updating existing facts. Further, application programs depend upon the DFB for all local or remote information exchange. Applications do not transmit information to other applications or DFBs directly; rather, information is entered into the local factbase and distributed to: one, other hosts when distribution rules are fired, or two, application programs when triggers are fired.

Local DFB queries and triggers constitute the only information gathering methods available to application programs. This is a radical change from the typical way of handling information, but allows a defined set of distribution rules to regulate information exchanges. However, if the users wants to send a free−text character string, this can be easily facilitated (although discouraged). Hopefully, combat developers and military scientists will eventually have data abstractions to cover all situations so that free−text messages are rarely, if ever, required.

The application programs implemented for the initial experimental IDS serve two main functions: to manually manipulate facts (i.e., create, update and kill facts) and to convert facts into a form understandable by the user (preferably in a graphical form). One simple purpose of application programs is to serve as an interface between the user and a DFB. Through applications, the user may display or enter DFB facts. Since facts are data abstractions of military concepts in their "purest" form, they must be converted into a form appropriate for human assimilation.

The implementation experience of the application developers suggests that any difficulty in displaying a fact correlates directly to how "correctly" the data abstractions of the military concepts are represented. That is, if the data abstractions are

correct, then it is easy to develop ways to display that information, and vice versa. Many commonly used terms and phrases, such as ”target” and ”border”, have more subtle (and often simpler) meanings than superficial intuition affords. The current set of fact−types are constantly being scrutinized and revised based on implementation experience; fortunately, they are getting simpler and more general.

Four applications programs were developed for the Smart Weapons Systems (SWS) Demonstration to supports the preparation, maintenance, and dissemination of the information associated with a fire support plan in a maneuver operations order (OPORD). The creation and maintenance of an OPORD proved to be an excellent vehicle on which to base the development of the system's capabilities (especially when a dynamic, real−time environment is provided).

In order to portray real−time combat situations accurately, a *Scenario Driver* application was developed (see **Figure 11**). The scenario driver converts tactical input data into facts and enters them into a separate factbase for dissemination to command and control nodes being exercised. Thus, in the demonstration, the scenario driver served as many non−exercised (simulated) units in a typical US brigade.

The tactical inputs for the scenario driver were developed under contract and provided an unclassified, 60−minute, Fulda Gap battle between a friendly mechanized infantry brigade task force and an enemy tank division with one minute resolution down to the platoon level [5]. From a master event list (i.e., the “scenario”),

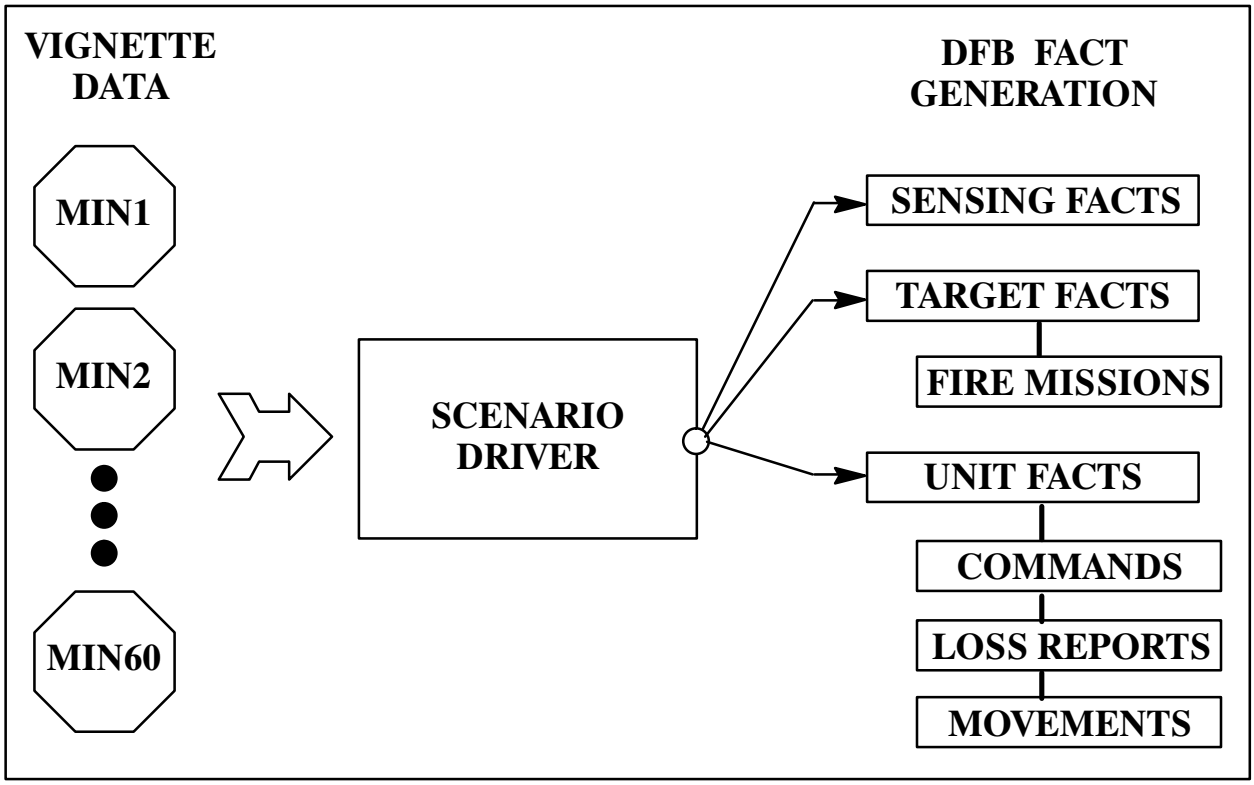

FIGURE 11: Scenario Driver Provides Fact−type Conversions

events are selected to build a “vignette”, that is, a particular blue perception of the battle; many vignettes can be built from the master list.

The *Organization Facts* (ORGCHART) application displays unit and Tables of Organization and Equipment (TO&E) related facts using common ”organization chart” diagrams. This application provides the ability to modify (i.e., task organize) the structure of military units and to display status roll−ups of both actual units and TO&Es for vehicles, equipment, ammunition, and personnel. The organization of any unit within the battlefield command structure may be examined via graphical output. Using the interactive work panel supported in ORGCHART, commanders and staff may attach subordinate units to build their task organization (changes in unit fact attached links). The current status of unit vehicles, equipment, and ammunition is provided through a ”roll−up” procedure. A ”roll−up” is the compilation of all the items assigned to the subordinate units of a unit under examination. A tabular listing is also provided that compares the current status with original unit conditions.

The *Working Map* (WMAP) application displays and manipulates geographical information detailing location and other attributes (e.g., range fans) of friendly and enemy units, sensings, geographic lines (one, two, and three dimensions), unit borders, and target facts. All graphical output can be superimposed onto a topographic map background, currently of the Fulda Gap area in Germany. The WMAP interactive panels allow a user to create geographical line, sensing, and target facts and to modify unit locations. Further, special features, such as the ability to generate a range fan for any weapon within a unit, are offered. Finally, “what_if” (meta) facts can be stated to discuss potential modifications of existing facts. This supports a “conversational graphics” capability by allowing the exchange of proposed unit actions and movements between DFBs (i.e., nodes) in graphical form. This capability helps alert planners to real−estate and other operational conflicts, in real time, during mission planning activities even though these personnel are geographically dispersed over the battlefield. **Figure 12** is an actual screen dump of a Sun computer display running both a WMAP and ORGCHART application program.

The *OPORD Builder* (OPLAN) application supports the display and creation of dynamic fire support plans and operations orders (OPORDs) using the *mission* fact−type. This fact−type contains much of the information typically associated with an operations order. The design of the data abstractions of the concepts associated with OPORDs was by far the most complex and allusive to date and is still ongoing. During the development of the mission fact definition, many postulated mission concepts were expressed in terms of already existent fact definitions. For example, a unit’s objective, route, and control measures can be expressed as a series of line facts entered graphically using the Working Map application. A unit’s task organization can be represented as a list of unit fact ids and be built using the ORGCHART application. As a result, OPORD generation and presentation by the OPLAN application became naturally dependent upon both the ORGCHART and Working Map applications.

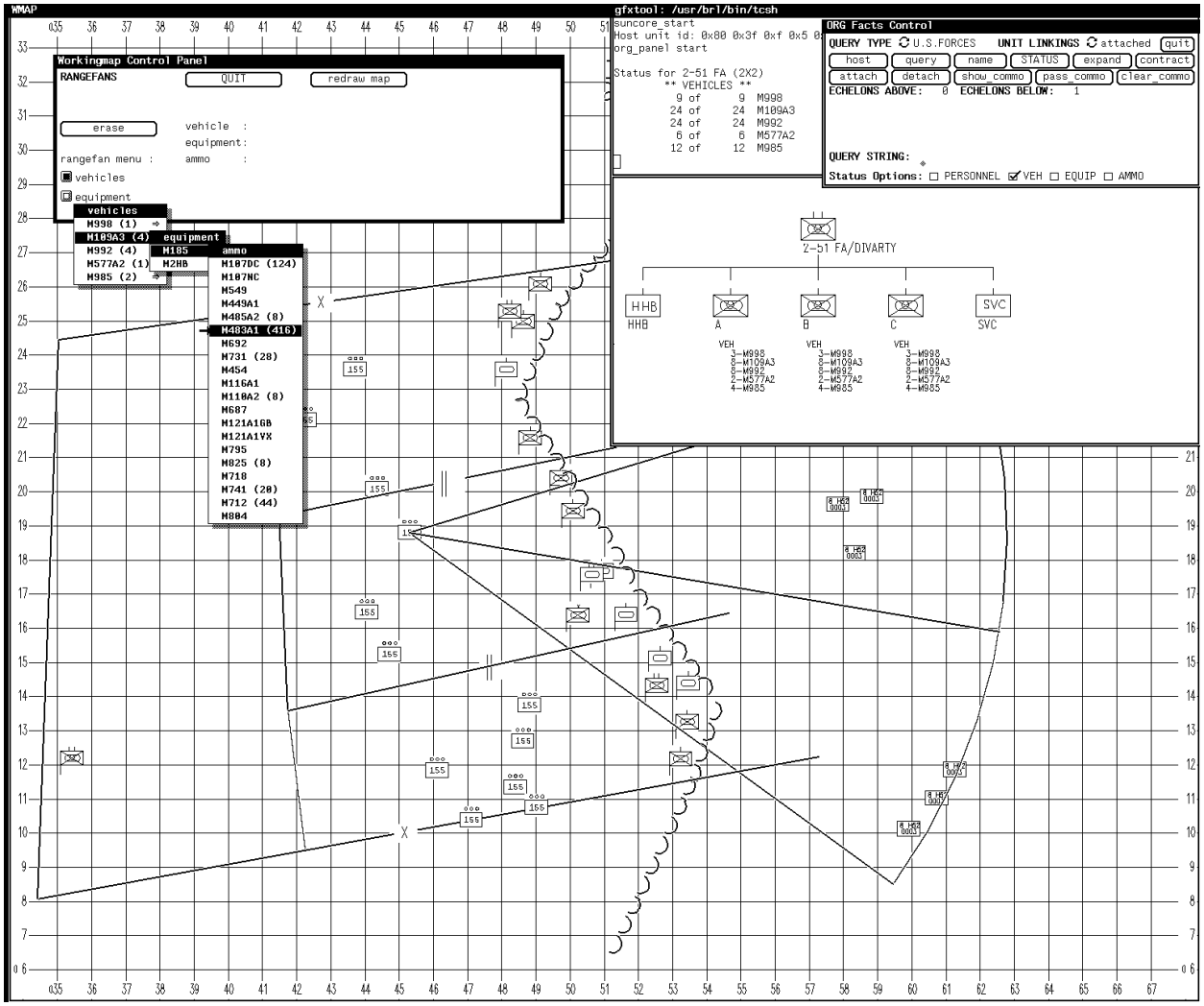

FIGURE 12: The Working Map and Org Chart Application Programs

In order to facilitate the compilation of information within the scope of a unit's mission, the development of an inter−application communication system became necessary. A new meta−fact, called a *commo* fact, is used to exchange fact ids between application programs via the DFB. Since only fact ids are being exchanged, very little bandwidth is required. This is a very simple inter−process communications scheme, but because it uses the DFB, it can be used for inter−process communications between different DFBs even when they are connected via low bandwidth communication channels. Much of the information required to build and display an operations order is created and manipulated using the ORGCHART and WORKING MAP applications in conjunction with commo facts. However, an interactive OPLAN graphics panel supports the display and building of OPORDs and the input of additional information, such as mission starting and ending times. Thus, the OPLAN application

provides a common ground for the association of otherwise disjoint information generated by other applications during the creation of an OPORD.

All of the applications use the Sun Microsystem workstation's graphics capability extensively for both data display and input. These application programs were used together to demonstrate the aforementioned DFB features and to assist the soldier by: identifying incoming information and alerting an operator, extracting current situation information from the factbase, graphically depicting unit mission and situation information, insuring that appropriate information is in the fact base, controlling the dissemination of fire support plan information, updating the prescribed CAPs, and making maximum use of graphics "tools" and other software available from the various SWS components.

IX. THE SWS PROGRAM

The IDS was used in the Smart Weapons Systems (SWS) LABCOM Cooperative Program, which included a command post exercise (CPX) and demonstration from 5 to 29 September 1989. The IDS provided the underlying information and communication infrastructure for two different suites of application programs.

The first suite included four sophisticated application programs developed by other SWS participants to execute the SWS fire missions. These applications were:

1. The *Information Processor* (IP) developed by Harry Diamond Laboratories (HDL), Adelphi, Maryland [9];
2. *FireAdvisor* developed by the Ballistic Research Laboratory [8];
3. The *Smart Howitzer Automated Management System* (SHAMS) developed by the Human Engineering Laboratory (HEL), Aberdeen Proving Ground, Maryland [10]; and,
4. The *Commander's Intelligent Display* (CID) developed by the Electronic Technology and Devices Laboratory (ETDL), Ft. Monmouth, New Jersey[7].

The second suite includes the four applications that are described in the previous section (i.e., ORGCHART, WMAP, OPLAN, and the Scenario Driver). These applications were used to modify an operations order in response to unexpected battle conditions presented by a scenario developed for this purpose. In addition, the DFB was expanded to include a *monitor* capability so that it could display message traffic between the nodes. Although this appears superficially to be an application program, it is actually part of the DFB software and is the beginning of a DFB control panel that will be expanded in the future.

Figure 13 illustrates the physical architecture of the eleven DFB nodes of the SWS demonstration. Dashed lines indicate separate geographical locations that were connected via radio links, with the exception of the scenario driver that operated totally over a LAN that connected all the nodes. DFBs are denoted with ovals, application programs as rectangles, DFB application interfaces as bold lines, and inter–DFB communications channels as solid lines. The communications channels are labeled as either high–speed LANs or VHF–FM radio, of which there were two.

As Figure 13 shows, there were seven separate locations containing the eleven

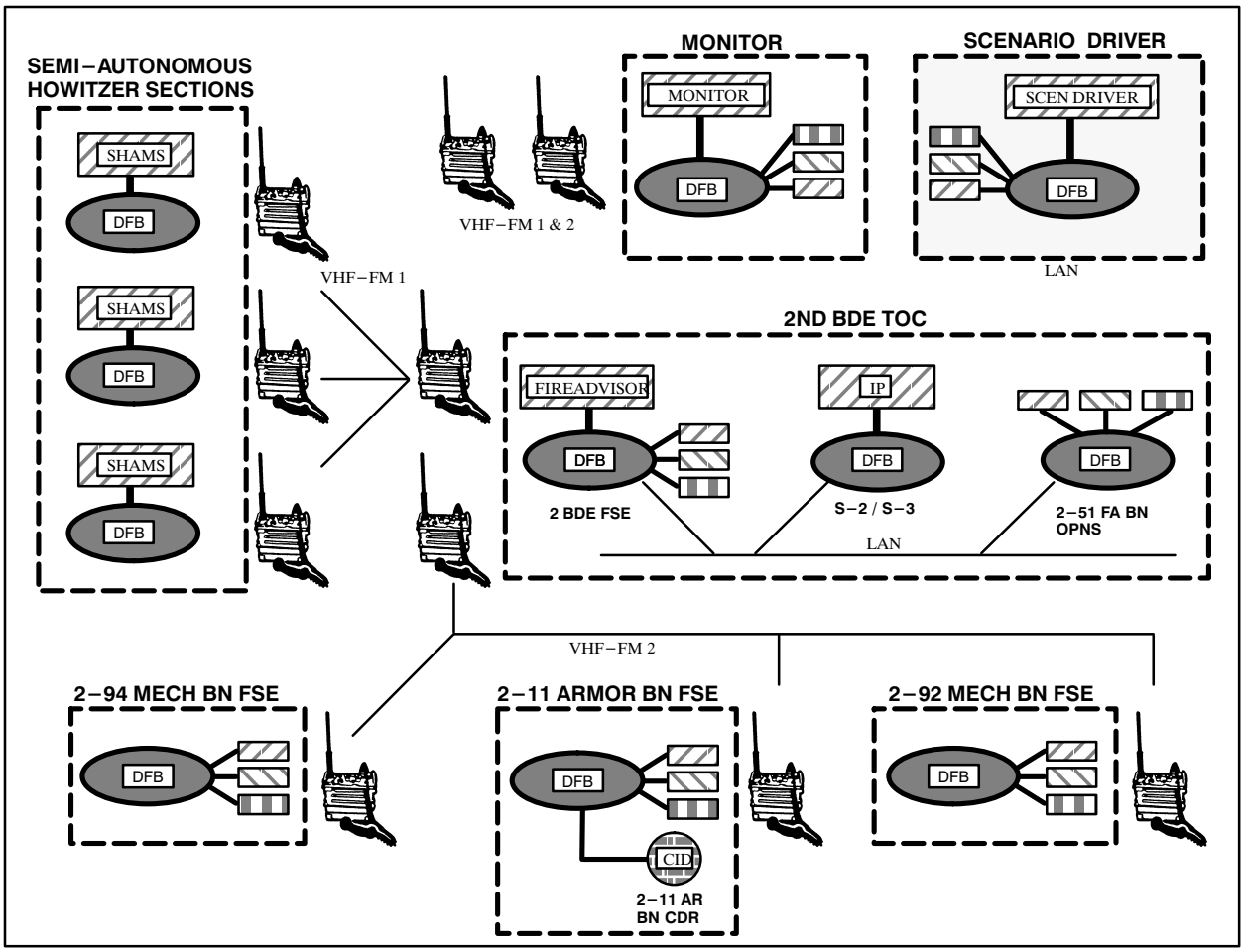

FIGURE 13: Physical Layout of the SWS Demonstration

DFBs. The scenario driver and monitor DFBs were located in the machine room and control room, respectively, and served support roles for the demonstration. The monitor DFB passively ''listened'' to the two VHF−FM radio links and the LAN. It collected the message traffic and allowed the operator to view the traffic from any combination of these networks.

The two suites of application programs were logically separated using distinct sets of IDCs (i.e., distribution rules) and separate VHF−FM channels. The first application suite used the ''VHF−FM 1'' channel to connect the brigade (bde) fire support element (FSE) with the FireAdvisor application to the three individual SHAMS DFBs. The Information Processor (IP) served the bde S−2/S−3 element, and since it was co−located with the bde FSE in the brigade tactical operations center (TOC), these DFBs were connected via a LAN.

The second suite of applications was used by the bde FSE, the three battalion (bn) FSEs, and the direct support field artillery battalion operations element (FA Bn OPS), each with their own DFBs. The ''VHF−FM 2'' channel connected the four geographically distributed FSEs, and since the FA Bn OPS element was co−located with the bde FSE in the bde TOC, this DFB also used the LAN in the TOC.

SWS MISSION PROFILE: FIRE MISSIONS

The focus of the SWS Demonstration was on the first suite of applications that supported SWS fire missions. The basic mission profile for a SWS mission follows. A scenario generator (not the one in Figure 13 and described previously) periodically entered simulated target spottings from a fictious remotely piloted vehicle (RPV) into the Information Processor (IP). The IP collected these individual sensings and collated them into groups worthy of attack. It also determined potential windows of engagement based on terrain and predicted when these target groups would enter the windows. The IP entered this information into its DFB where distribution rules fired to send this and other information to the bde FSE DFB (via the bde TOC LAN) for further processing by the FireAdvisor application program.

The FireAdvisor application program had triggers set in its DFB so that it was notified when potential targeting information arrived in the DFB. It maintained a list of sensings and their predicted arrival times in attack windows and determined which of the sensings were worthy artillery targets. Shortly before firing time, it entered selected artillery targets into its DFB. A distribution rule in the DFB fired and broadcasted (i.e., the fact exchanges were addressed to anyone) these targets on the VHF–FM 1 radio channel to be received by the individual SHAMS sections.

Each SHAMS application program had triggers set in its DFB so that it was notified when fire missions arrived in the DFB. In response to the trigger, the SHAMS automatically retrieved the fire mission information, which did not yet include a unit to fire, and calculated a ''bid'' value based on its past and projected operational assignments. This included ranging capabilities, ammunition availability, vulnerability to enemy attack (e.g., number of recent missions fired with charge and quadrant), and pending preplanned missions. This bid was entered into its DFB where a distribution rule fired sending it back to the bde FSE DFB with the FireAdvisor application program.

After waiting a predetermined amount of time for bids to return (e.g., 60 seconds), the FireAdvisor program queried its DFB for bids and retrieved them for processing. Based on the bids and other parameters, it selected a unit to fire, and upon authorization by the operator (if desired), updated the target information in its DFB. A distribution rule fired sending this update to the selected SHAMS section DFB for execution. The SHAMS had a trigger set for this type of information, and upon retrieving the fire mission assignment, notified its crew who executed the fire mission.

In all cases, the firing of distribution rules and triggers was automatic within a DFB. It was up to the users, via the application programs, to determine the level of operational automation that was desired. For example, in the previous mission profile, the SHAMS section crews never knew the bidding process was occurring. They were only notified once the section had been selected to fire the mission.

SWS MISSION PROFILE: PLANNING

The tactical theme of the original IDS proposal [1] revolved around the dynamic actions of five key field artillery nodes (the

maneuver brigade fire support element, a direct support field artillery battalion operations element, and three maneuver battalion fire support elements) that are responsible for two key fire support functions: *fire support control* & *coordination* (FSCC) and *field artillery tactical operations* (FA TAC OPS). In 1987, the additional nodes were added to support the SWS program orientation towards a goal of second echelon attack for brigades. However, this new goal quickly took precedence over the original IDS objectives.

The primary purpose of the second suite of applications was to support the development and evaluation of the DFB and the data abstraction of military concepts. A second objective was to use this suite in the SWS Demonstration pending success of the first suite of applications and time permitting. This was partially completed since considerable traffic was exchanged between the bn and bde FSEs using the ORGCHART and WMAP applications; however, full replanning was never exercised using the OPLAN application. Although this second objective was not completed, the development of the OPLAN application was extremely successful and enlightening. Consequently, replanning and OPORD processing will remain an operational evaluation medium of future technology efforts.

SWS DEMONSTRATION

During the exercise, the traffic exchanged between nodes (DFBs) was collected as well as data on the information exchanged between the application programs and the DFBs. Due to its size and complexity, the SWS Demonstration was not attempted as an experiment (i.e., no experiment design plan was developed); therefore, the data can not be rigorously analyzed and valid conclusions formed. However, the data reflects a significant increase in efficiency and the usefulness of the DFB concepts, the FEP features (e.g., overhearing), and the appropriateness of the data abstractions. In addition, subjective information was obtained from the operators concerning their ability to use the applications and to understand the situation that was presented.

Communications among the eleven nodes were recorded and examined during simulations of various combat situations. Tactical vignettes were employed to create critical combat situations [5]. The vignette information was converted into facts by the scenario driver. The scenario driver entered facts into its own separate DFB concerning the movements and status of 172 simulated blue units and their enemy contacts, engagements, and spottings (as described in the vignettes). Distribution rules immediately fired sending this information to the FSEs and the DS artillery battalion S−3, which in turn, passed information to the other DFB and their associated application programs. The scenario driver communicated via the LAN since this part of the exercise was simulated.

The performance of the DFB was outstanding. The DFB that serviced the scenario driver was the most heavily loaded DFB. It successfully handled over 20,000 factbase commands per hour, and this included processing the IDC for each incoming command. The FEP worked well over both the LAN (as expected) and over standard PRC−77 VHF−FM radios. The broadcast and multicast schemes significantly

decreased bandwidth utilization for fire mission processing since a single message was received by several hosts. The triggering mechanism quickly and efficiently alerted application programs when important information arrived. Finally, the data abstractions were highly successful in providing a primitive data format that proved efficient for both data transmission over low−bandwidth communication systems and processing by the sophisticated application programs using them.

X. CONCLUSION

The underlying goal of this project was to develop an experimental information distribution system that was still preferred by the user during degraded modes of operation. Through the careful consideration of both military and computer science concepts, a canonical list of information primitives was derived. It is hoped that every type of military concept will eventually be represented by a data abstraction rather than text. As the developers gained more insight into the complex tasks of command and control, the list and contents of the primitive fact−types continued to become simpler and more general.

The basic concept of providing a computationally intensive system rather than a communications intensive one proved demonstrably viable and valid. This is especially significant when low−bandwidth tactical radios are the means of communications. The Fact Exchange Protocol performed well under these conditions due to its capability to broadcast and overhear, and its terse exchange format. These features were exploited by the application programs; this was most notably demonstrated by the new method of fire mission processing via negotiation. The DFB trigger mechanism allowed the application program users to focus on their primary tasks and not constantly check the DFB for information updates. Finally, the architectural concept of separating information distribution tasks from applications proved exceptionally useful. This was especially noteworthy since it provided a clean break between the software and hardware required for the common task of information distribution and the highly specialized requirements often encountered by application programs.

It is further believed that automatic information distribution becomes more important the *lower* one goes down the fighting echelons, see **Figure 14**. Combat system operators such as armored vehicle crews as well as foot soldiers are far too busy fighting and surviving to type information into keyboards. It is imperative that automated systems provide relief from the many mundane tasks associated with information distribution. But this must be accomplished in a realistic manner with the limited bandwidth communications systems required for highly variable terrain (e.g., non−line−of−sight conditions).

Hopefully, new information distribution technology concepts such as the Distributed Factbase, Capability Profiles, the features of the Fact Exchange Protocol (multicast, overhearing, and listening silence), and the application programs when combined with carefully developed data abstractions of military concepts will provide responsive operations in spite of severely limited communications. If not, the equipment will be thrown aside during crisis situ-

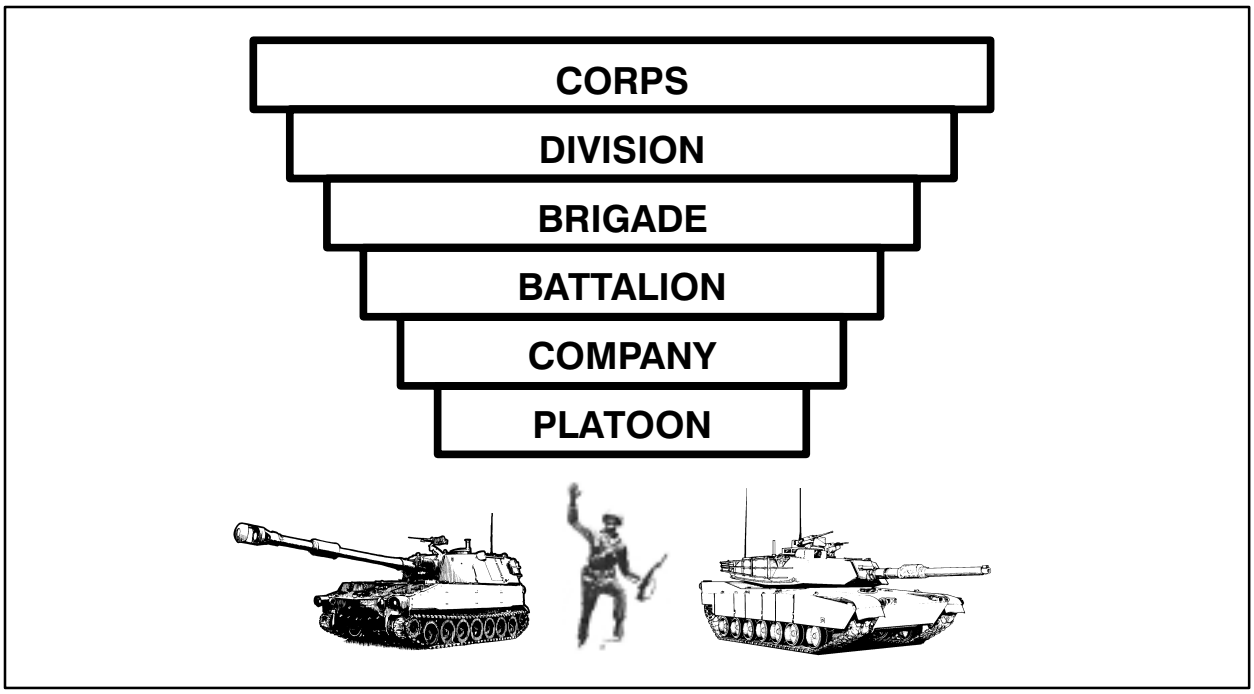

FIGURE 14: Automatic Distribution Requirements – Most Important at Lower Echelons

ations, and commanders and their staffs will be forced to huddle in circles using fig-ure drawings in the dirt to plan and control the battle.

ACKNOWLEDGEMENTS

This work evolved over the past three years and is the result of the synergistic effect of the combined contributions of several members of the BRL's System Engineering and Concepts Analysis Division (past and present); those members are: Tom DiGiacinto, Phil Dykstra, George Hartwig, Eric Heilman, Ginny Kaste, Don Merritt, Mike Muuss, Joe Pistritto, Ken Smith, Wendy Winner, and Mike Zoll.

REFERENCES

[1] S. C. Chamberlain, et al.,''The BRL Fire Support Application For ADDCOMPE,'' Ballistic Research Laboratory Special Report No. BRL−SP−53, April 1986.

[2] DDN Protocol Handbook (NIC 50004) − Vol One, DoD Military Standard Protocols, DDN Network Information Center, SRI International, Menlo Park, CA 94025, December 1988.

[3] Information Processing Systems − Open Systems Interconnection: Basic Reference Model, International Standard 7498.

[4] K. Knightson, et al., *Standards for Open Systems Interconnection*, McGraw−Hill, New York, N.Y., 1987.

[5] Tactical Inputs for Fire Support Control, Contract Number DAAA15−85−C−0107 for US Army Ballistic Research Laboratory, by LB&M Associates, 211 SW A Avenue, Lawton, OK 73501−4051, 1985−1986.

[6] H. Zimmermann, ''OSI Reference Model − The ISO Model of Architecture for Open Systems Interconnection,'' *IEEE Trans. on Communications, COM−28 (4)*, April 1980, 425−432.

Other Smart Weapons Systems (SWS) LABCOM Cooperative Program Reports
(in Publication)

[7] D. Chiu and E. Tuttle, ''Smart Weapons Systems/Commander's Intelligent Display (SWS/CID) Final Report,'' Electronic Technology and Devices Laboratory technical report.

[8] R. C. Kaste, ''An Experimental Artillery Decision Aid: Concepts and Implementation,'' Ballistic Research Laboratory Technical Report.

[9] L. Tokarcik, ''The Information Processor: IP − Real Time Prediction and Tracking,'' Harry Diamond Laboratories technical Report.

[10] O. Zubal and M. Thomas, ''Development and Evaluation of a Smart Howitzer Automated Management System (SHAMS),'' Human Engineering Laboratory Technical Report.

LIST OF ABBREVIATIONS, ACRONYMS, and TERMS

APFSDS−T	Armor Piercing, Fin Stabilized, Discarding Sabot with Tracer (ammunition)
BN	Battalion
BPS	Bits Per Second
BRL	Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland
CAL	Caliber
CAP	Capability Profiles
CID	Commander's Intelligent Display (built by ETDL)
COMMO	Commo fact
CP	Command Post
CPX	Command Post Exercise
DEST	Destination (of a message)
DFB	Distributed Factbase
D−SDU	Datalink Layer Service Data Unit
DoD	Department of Defense
ETDL	Electronic Devices and Technolgy Laboratory, Ft Monmouth, New Jersey
EMCON	Emission Control
FA TAC OPS	Field Artillery Tactical Operations
FEP	Fact Exchange Protocol
FEX	Fact Exchange
FID	Fact Identification Number
FSCC	Fire Support Control and Coordination
FSE	Fire Support Element
FSK	Frequency Shift Keying
HDL	Harry Diamond Laboratory, Adelphi, Maryland
HDR	Header (of a message)
HEAT−T	High Explosive, Anti−Tank with Tracer
HF−AM	High Frequency − Amplitude Modulation
HEL	Human Engineering Laboratory, Aberdeen Proving Ground, Maryland
IDC	Information Distribution Command
IDS	Information Distribution System
INTEL	Intelligence
IP (1)	Internet Protocol
IP (2)	Information Processor (built by HDL)
ISO	International Standards Organization
LABCOM	US Army Laboratory Command, Adelphi, Maryland
LAN	Local Area Network
MBT	Main Battle Tank
MTU	Maximum (Size) Transmission Unit

N−PDU	Network Layer Protocol Data Unit
N−SDU	Network Layer Service Data Unit
OPNS	Operations
OPORD	Operations Order
ORG	Organization fact
OSI	Open Systems Interconnection
PCI	Protocol Control Infromation
PDU	Protocol Data Unit
PLT	Platoon
RMOSI	The Reference Model of Open Systems Interconnection
RPV	Remotely Piloted Vehicle
S−2	Intelligence Staff Section/Officer
S−3	Operations Staff Section/Officer
SASE	Special Application Service Element
SCM	Security Control Module
SDU	Service Data Unit
SHAMS	Smart Howitzer Automated Management System (built by HEL)
SOP	Standard Operating Procedures
SWS	Smart Weapons Systems
TCP	Transmission Control Protocol
TOC	Tactical Operations Center
TO&E	Table of Organization and Equipment
T−PDU	Transport Layer Protocol Data Unit
T−SDU	Transport Layer Service Data Unit
UDP	User Datagram Protocol
VEH	Vehicle fact
VHF−FM	Very High Frequency − Frequency Modulation
WSTB	Weapons Systems Technology Branch (of BRL)